Effects of Context-Sensitive Delays on Group Dynamics in 3D Virtual Worlds

Abstract
3D online virtual worlds are platforms for cooperative applications. They allow users to create customized content. Changes in the system conditions are usually manifested to end users as delays in the shared state. Consistency management and delay compensation solutions for other collaborative environments may not be directly applicable because of context-sensitive changes in the system behavior. Our study of a social dilemma task explores the extent to which cooperation dynamics are affected by network and context-sensitive delays. The results show that context-sensitive delays led to accidental damage of the collective action.

Keywords
3D online Virtual Worlds, Group Dynamics, Social Traps

Introduction
3D online Virtual Worlds, such as Second Life (SL), are platforms for cooperative applications with no Quality of Service guarantees. Determining system requirements
to provide acceptable Quality Of Experience (QoE) is an open research problem. Requirements vary according to the number of users and their activities including the content they create and customize [1]. Changes in the system performance are context-sensitive and non-linear. A small change in the number of users or their activities could result in larger changes in the perceived QoE. SL is a client-server architecture where the server is responsible for maintaining the consistency of the shared state. Context-sensitive server delay may occur due to executing the scripts for customized user objects. However, the server does not have enough information as to how the delay in the object script would affect the consistency of the shared state. Context-sensitive server delays are neither predictable nor controllable by the system designers.

The solutions where a system maintains QoE are still unclear, both at the interface and architecture levels. We need to understand how context-sensitive changes in the system affect users and their QoE. We focus on cooperative tasks and the dynamics of cooperation. We conducted a study where groups of strangers performed a task that simulates social traps. Social trap is a situation that resembles a group that possesses a common resource. It is in the individual’s best interest to selfishly consume a common resource to achieve fast short-term gain. However, if all the individuals act selfishly, the group ends up damaging its resource in the long run. We designed the task with theoretically sufficient conditions for cooperation to emerge and be sustained. We examined the sustainability of cooperation in the presence of network and context-sensitive server delays. We classify groups according to their responses and adaptation to system changes.

**Study**
The first laboratory experiment to simulate social traps was conducted by Kevin Brechner in the 70’s. Participants played a game in groups of three to draw points from an “artificial” common pool automatically replenished at fixed rates. The replenishment rate varies according to the number of points exist in the pool. The higher the number of points the faster the pool is replenished. If the participants consumed the points faster than the replenishment rates, the pool would empty and the game ends. Each participant should draw as many points from the pool as possible. If the pool was depleted before the original game time passes, the game ends immediately.

![Figure 1: Social Traps in Second Life](image)

The social traps experiment was replicated in SL [2], where participants’ behavioral patterns were found to be similar to the original experiment by Brechner. The
presented study is a modification of the replicated experiment, where the updates of the shared state were delayed. The pool was represented using a visual display made of 24 small balls arranged in a vertical line (Figure 1). The red ball marks the total number present in the pool at a given time. Each time a point is drawn from or added to the pool, the red ball moves down or up, respectively. Each participant was given a counter (the black box in Figure 1). One click of the counter adds one point to the participant’s counter and subtracted one from the pool. Participants can see their score on the local chat as a message sent by their counters each time the counter is clicked. Participants cannot see each other’s scores.

We had 15 groups. Each group played the games for six sessions. Participants were placed in three separate rooms. The only interaction they had was via SL. Participants communicated using text chat messages. Neither the number of sessions, nor the duration of each session were known to the participants so they could not plan to deplete the pool at the end of the game or session. Depletion of one session terminates only that session. Participants were compensated in course credit hours (1 to 4 hours) based on their score.

A local OpenSimulator server was used to have full control over the delay between the server and clients. Server and clients were connected on a LAN. Clients used SLViewer2 to connect to the server. We tested two types of delays (Figure 2). The duration and delay type for each session is shown in Table 1. Network delay is added at the network level between the client and the server. The context-sensitive server delay was introduced in the script that updates the red ball position. The chat message indicating the updated user score was not delayed which introduced inconsistency in the presented information. The local chat for each participant indicates new points added to her score. The red ball position, however, does not match the current number of points in the pool.

![Figure 2: Network Delay and Context-Sensitive Delay](image)

<table>
<thead>
<tr>
<th>Time</th>
<th>Delay Type and Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No delay</td>
</tr>
<tr>
<td>2</td>
<td>Gradually increasing network delay (Round Trip Time 1 to 5 secs)</td>
</tr>
<tr>
<td>3</td>
<td>Gradually increasing context-sensitive delay (0 to 4 secs)</td>
</tr>
<tr>
<td>4</td>
<td>Constant context-sensitive delay (1 sec)</td>
</tr>
<tr>
<td>5</td>
<td>Constant context-sensitive delay (3 secs)</td>
</tr>
<tr>
<td>6</td>
<td>No delay</td>
</tr>
</tbody>
</table>

Collected Data
We collected for each participant her responses (clicks) per second or the rate at which points were consumed from the pool. At the group level, we measured the number of replenished points as an indicator for the group ability to regulate the pool. We video recorded participants’ screens and stored their chat logs.

Findings, Conclusions and Future Work
All the groups successfully avoided depletion in the network delay sessions. Eight out of the 15 groups
avoided depleting the pool in context-sensitive delay sessions. For the remaining groups, depletion occurred twice in session 3, once in session 4, sixth times in session 5. For two groups depletion occurred in sessions 3 and 5.

![Figure 3](image)

Figure 3: Classifying Group Responses to System Changes

In the post-experiment questionnaire, participants were asked to rank their group members on a five points Likert scale from one (non-cooperative) to five (highly cooperative). We found a negative statistical correlation between this rank and the number of depleted sessions $[\text{Spearman's } \rho = -0.6667, p < 0.0001]$. Adaptation to network delays was easier than adapting to context-sensitive delays.

We observed noticeable similarity in the clicking patterns between groups that depleted the pool and those who avoided depletion. Examining the videos for depletion groups does not suggest that depletion was intentional. We further analyzed the chat and classified groups according to how they shared their observations and responded to the delays (Figure 3). Depleted groups are shown using red font. Depletion groups appear consistently in each category, which is another indication that depletion happened accidentally.

Except for two groups, all groups observed the changes in the system and shared their observations. The knowledge that delay exists was not sufficient to avoid depletion. Groups were reactive in addressing the change in the system. The qualitative data suggests that group adaptation to the changes was mostly an aggregate of individual adjustments rather than a plan agreed on by members. Adopted plans were as general as “slow down”, “be careful gys”, or “let it refill”.

Successful responses in sessions 3 to 5 did not necessarily reflect a correct model for the delay.

We plan to overlap our findings from the quantitative and qualitative data analysis to characterize patterns of behavior at the individual and the group level. These patterns would help understanding the adaptation to contextual changes at both levels. Such understanding would help assessing the feasibility of providing infrastructure solutions to maintain QoE under context-sensitive system changes.

References